Faster polynomial multiplication over finite fields

David Harvey

University of New South Wales

8th December
AustMS/NZMS 2014, University of Melbourne

(joint work with Joris van der Hoeven and Grégoire Lecerf, École polytechnique)

Given $F, G \in \mathbf{F}_p[X]$ of degree $< n$.

Want to compute $H = FG$.

How fast can we compute H?

Assume an *algebraic complexity model*: count the number of ring operations in $\mathbf{F}_p$, i.e., additions, subtractions, multiplications.

Let $\mathrm{M}(n)$ be the number of operations needed to multiply polynomials of degree $< n$.

Want to design algorithms that minimise $\mathrm{M}(n)$ asymptotically.

(An alternative measure would be *bit complexity* — number of Boolean operations on single bits. This takes into account the size of p.)

Primary school long multiplication algorithm:

$$\mathrm{M}(n) = O(n^2).$$

Multiply every coefficient of F by every coefficient of G, then add.

The best known bound prior to our work was an algebraic version of the Schönhage–Strassen algorithm (1971), based on fast Fourier transforms:

$$\mathrm{M}(n) = O(n \log n \log\log n).$$

In fact this bound holds over any algebra, not necessarily commutative or associative (Cantor–Kaltofen, 1991).

Theorem (H.–van der Hoeven–Lecerf, arXiv July 2014)

We have the bound

$$\mathsf{M}(n) = O(n \log n \, 8^{\log^* n}).$$

Here $\log^*$ is the iterated logarithm:

$$\log^* x = \begin{cases} 0 & \text{if } x \leq 1, \\ 1 + \log^*(\log x) & \text{if } x > 1. \end{cases}$$

It is a *very* slowly growing function!!! Example:

$$\log^*(e^{e^{e^{e^e}}}) = 5.$$

Fürer proved an analogous bound for *integer* multiplication in 2007.

But Fürer's method does not appear to work for polynomials.

The new method works for both integers and polynomials. Today I will give a brief outline of the polynomial case.

Step 1

Construct a suitable extension of $\mathbf{F}_p$.

We want a "small" extension $\mathbf{F}_q$ of $\mathbf{F}_p$ (say $q = p^\lambda$) that contains "many" roots of unity of "small" order.

In other words, we want to find a "small" positive integer λ such that $p^\lambda - 1$ has "many" divisors that are "small".

Idea: if r is a prime ($\neq p$) such that $r-1$ divides λ, then r divides $p^\lambda - 1$.

Define

$$H_\lambda := \prod_{\substack{r \text{ prime}, \\ r-1|\lambda}} r.$$

We want H_λ to be *large*.

In other words, we want λ to have many divisors d such that $d+1$ happens to be prime.

Example:

$$H_{60} = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdot 31 \cdot 61.$$

This explains why

$$\begin{aligned}
2^{60} - 1 &= 3^2 \cdot 5^2 \cdot 7 \cdot 11 \cdot 13 \cdot 31 \cdot 41 \cdot 61 \cdot 151 \cdot 331 \cdot 1321 \\
3^{60} - 1 &= 2^4 \cdot 5^2 \cdot 7 \cdot 11^2 \cdot 13 \cdot 31 \cdot 61 \cdot 73 \cdot 271 \cdot 1181 \cdot 4561 \cdot 47763361 \\
5^{60} - 1 &= 2^4 \cdot 3^2 \cdot 7 \cdot 11 \cdot 13 \cdot 31 \cdot 41 \cdot 61 \cdot 71 \cdot 181 \cdot 521 \cdot 601 \cdot 1741 \\
&\quad \cdot 2281 \cdot 7621 \cdot 9161 \cdot 69566521
\end{aligned}$$

are divisible by many small primes.

How do we construct λ in general?

The heavy lifting is done by the following analytic number theory result, due to Adleman–Pomerance–Rumely (1983).

Theorem

There are absolute constants $c > 0$ and $n_0 \geq 1$ such that for any $n \geq n_0$, there exists some

$$\lambda \leq (\log n)^{c \log\log\log n}$$

such that

$$H_\lambda \geq n.$$

Notice that λ is *exponentially smaller* than n, but we still manage to force $p^\lambda - 1$ to be divisible by a λ-smooth integer of size comparable to n.

Step 2

Reduce multiplication in $\mathbf{F}_p[X]$ to multiplication in $\mathbf{F}_q[Y]$.

Given $F, G \in \mathbf{F}_p[X]$ of degree $< n$.

Construct an extension $\mathbf{F}_q$ as in step 1.

Represent $\mathbf{F}_q$ as $\mathbf{F}_p[Z]/P$ for some irreducible $P \in \mathbf{F}_p[Z]$ of degree λ.

Cut up F and G into $\approx 2n/\lambda$ chunks of $\lambda/2$ coefficients each.

Encode each chunk as an element of $\mathbf{F}_q = \mathbf{F}_p[Z]/P$.

Thus we reduce to multiplying polynomials in $\mathbf{F}_q[Y]$ (of degree $\approx 2n/\lambda$).

Step 3

Reduce multiplication in $\mathbf{F}_q[Y]$ to evaluation/interpolation over $\mathbf{F}_q$.

To multiply polynomials in $\mathbf{F}_q[Y]$, it suffices to

- choose suitable integer N,
- evaluate $F(\omega)$ and $G(\omega)$ for each N-th root of unity $\omega \in \mathbf{F}_q$,
- compute $H(\omega) = F(\omega)G(\omega) \in \mathbf{F}_q$ for each ω, and
- interpolate to recover $H(Y) \in \mathbf{F}_q[Y]$.

The evaluation is really just a discrete Fourier transform (DFT) over $\mathbf{F}_q$.

The interpolation is an inverse DFT.

How to choose N?

We need:

- $N \geq 2n$ (there are enough evaluation points),
- N divides $q - 1$ (the roots of unity exist), and
- $N = \prod_i N_i$ where the N_i are exponentially smaller than n.

Our construction of λ ensures we can choose such N.

Step 4

Reduce N-point DFT to many small N_i-point DFTs.

Use the Cooley–Tukey algorithm (the original "fast Fourier transform") to reduce each N-point transform to many small N_i-point transforms.

Details omitted.

(This is really just an algebraic rearrangement. It is a generalisation of the usual decimation-in-time and decimation-in-frequency FFT algorithms for sequences of length 2^n.)

Step 5

Reduce N_i-point DFT to polynomial multiplication of degree N_i.

Previously we reduced polynomial multiplication to computing DFTs.

In this step we use Bluestein's algorithm (1970) to go the other way.

Details omitted.

(Again this is just a clever algebraic trick.)

Step 6

Reduce multiplication in $\mathbf{F}_q[Y]$ back to multiplication in $\mathbf{F}_p[X]$.

Previously we reduced multiplication in $\mathbf{F}_p[X]$ to multiplication in $\mathbf{F}_q[Y]$.

In this step we use Kronecker substitution to go the other way.

Basically this involves "forgetting" the modulus $P(Z)$, and concatenating the coefficients with appropriate zero-padding.

Details omitted.

Step 7

Handle much smaller multiplications in $\mathbf{F}_p[X]$ recursively!

Overall this reduces multiplications of degree n to many small multiplications of degree exponentially smaller than n.

With some effort, this leads to the bound

$$\mathrm{M}(n) = O(n \log n \, K^{\log^* n})$$

for some $K > 1$.

With even more effort, one obtains $K = 8$.

Can we do better?

Theorem (H.–van der Hoeven–Lecerf, arXiv July 2014)

Assuming several conjectures, we can achieve

$$\mathrm{M}(n) = O(n \log n \, 4^{\log^* n}).$$

I will briefly explain one of these conjectures.

Let p be a prime.

Artin's conjecture states that there are infinitely many primes r such that p is a primitive root modulo r, and predicts their density.

It was proved by Hooley under GRH (1967).

Without GRH, there is not a single prime p for which it is known to hold.

What about "Artin's conjecture in arithmetic progressions"?

Let $A(p, \lambda)$ be the set of primes r such that p is a primitive root modulo r, and r lies in the arithmetic progression

$$r = 1 \pmod{\lambda}.$$

Lenstra proved the following (1977):

Theorem

Assume GRH. Let $\lambda \geq 2$ and let p be a prime. If p is odd, assume that $p \nmid \lambda$; if $p = 2$, assume that $8 \nmid \lambda$. Then $A(p, \lambda)$ is infinite and has the "right" density.

To achieve $\mathrm{M}(n) = O(n \log n \, 4^{\log^* n})$, we need an upper bound for the *smallest* element in $A(p, \lambda)$.

So it is basically like "Artin plus Linnik".

Of course we are willing to assume GRH!

We could get away with something quite weak... for example it would be enough to prove (under GRH):

Conjecture

Assume the same hypotheses as Lenstra's theorem. Assume also that $\lambda \geq \exp(\exp(\exp(\exp p)))$. *Then the smallest element of* $A(p, \lambda)$ *is less than* $\exp((\log \lambda)^{100})$.

We will be very happy if anyone can prove this!